International Journal of Computer Applications (0975 – 8887)
Volume 72– No.13, May 2013

Requirement Elicitation: A “Live” Prototyping Approach

Narayan Ranjan Chakraborty
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Subhenur Latif
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Yousuf Mahbubul Islam
Professor
Department of Computer Science and Engineering
Daffodil International University

ABSTRACT
Requirement gathering has traditionally been acknowledged as the most crucial as well as the most difficult step in the Software Development Life Cycle (SDLC). Over the years, feedback from customized software packages, availability of standardized software packages as well as documented business processes has helped the budding developer in understanding business needs. In Bangladesh, a developing nation, there are many small business enterprises that are looking to computerize some of their business information processes. As the volume of their transactions does not justify the purchase of expensive ERP solutions, they look for small-data-volume customized solutions. The article shares how using a “working” or “live” prototype has been useful in gradually specifying customer needs, eliciting feedback and freezing requirements from clients who are not fully conversant with the benefits of computerization.

Keywords
Prototyping, Requirement Engineering, Working Prototype, SDLC, Freezing Requirements.

1. INTRODUCTION

Finding the information needs, identifying business processes and specifying a client’s requirements are classified under Requirement Engineering. Gathering the requirements properly is most crucial as succinctly pointed out by Westfall [1] in her abstract, “If software requirements are not right, companies will not end up with the software they need.” The Westfall Team has therefore done some useful work in the area of Requirement Engineering. With the advancement and addition of a variety of automation tools that need to work with computers, *Requirement Specification* has become even more complex than ever before. For this purpose, Westfall [1] has proposed a useful taxonomy suggesting “Levels and Types of Requirements” as shown in Figure 1[1] below.

Westfall[1] herself adapted the levels from “Sub-disciplines of Requirements Engineering” as proposed by Wiegers. As seen in Figure 1, at the basic or Business Level the *Business Requirements* define the business information problems that need to be solved or the business opportunities that need to be addressed. In practice, collecting the basic Business

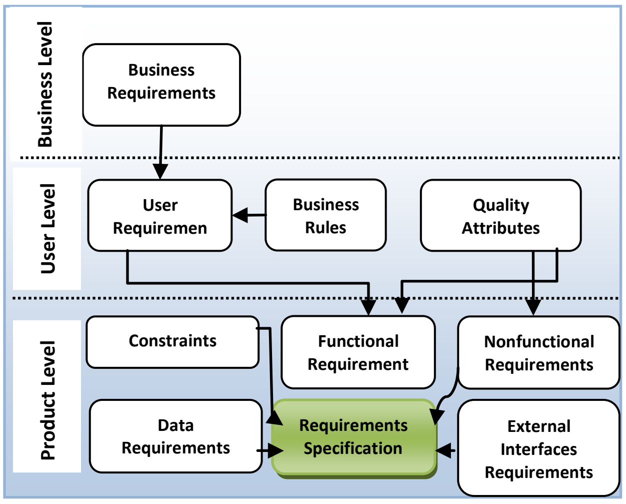


Figure 1: Levels and types of requirements

Requirements, or the *User Requirements* and *Business Rules* at the User Level prove to be most difficult. In Bangladesh, this is more so as users may be unfamiliar with the use of computers and some Business Rules may follow thumb-rules or rules that have never been documented. This is especially true for small companies that may be sole-owner or a partnership or even be classified as a Small Medium Enterprise (SME). In such cases, gathering requirements is difficult. A common response of those with little or no knowledge of computer processing is, “You-being-the-computer-specialist-know-best-what-to-do.” Without active participation of the user-client at the Business Level and subsequent User Level stages, it would be impossible to complete the final Requirement Specification for the Product Level as shown in Figure 1.

2. WORKING OR ‘LIVE’ PROTOTYPE

It is to elicit the participation of such a client in the process of Requirement Gathering, particularly at the Business Level and User Level that this paper has made use of a Working or Live Prototype. As part of a client’s documentation, there is usually some sort of ledger books where inventory or accounts are kept [7, 10]. To elicit participation, a spreadsheet is used to get the client-user to specify requirements in a step-by-step manner. The spreadsheet, in the first instance, is used to mimic the format of the ledgers that the user is so familiar with.

The prototype is called 'Working' or 'Live' as gradually menu options or buttons are added (using a third party Visual Basic for Applications, VBA) to the spreadsheet which allow the client-user to visually observe the processing that goes on to produce the required reports as well as add Data Entry Forms for inputting required data. The client-user can enter real data and press buttons as required to get and check the reports produced. As the real data is 'visible', the client can see and understand how the computer handles processing and give feedback as required. Working with data in this manner, the client is also able to 'quantify' any popular thumb rules used in the business processes as he/she is able to understand the need for formulae.

The next section describes the steps taken to finalize the Business Level and as well as the User Requirements and Business Rules at the User Level as described in Figure 1. Subsequent sections share a number of case studies in which this approach was applied. In each of these case studies MS-Excel was used as the spreadsheet with a Visual Basic for Applications (VBA) to introduce buttons and VBA code for automatic report production.

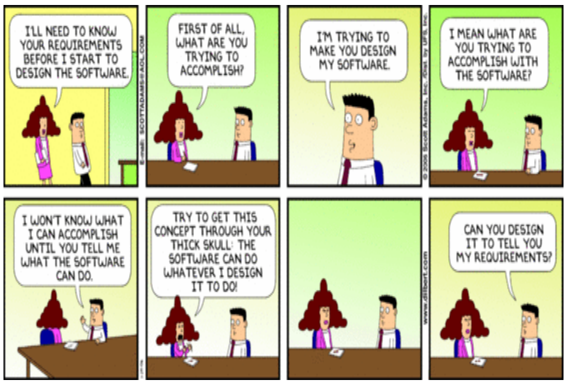

I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

Fig 2: A cartoon to show why Requirement Gathering is difficult in practice

3. LIVE PROTOTYPING METHODOLOGY

The Live Prototyping methodology has been divided into a series of specified steps or phases. Each step involves interaction that must be followed for successfully gathering requirements. The methodology is shown in Figure 3 in the form of a flowchart. The third author Islam, who is a practitioner, has adapted and improved this methodology as part of his Systems Analysis and Design course since 2004.

3.1 Pre-prototype

This is the self-preparation phase before the first data gathering visit to the client.

- ➢ The analyst or practitioner must first do his/her homework by first guessing what the proposed needs of the business may be. The following question should be answered: "What do I know about the information needs of such businesses?"
- ➢ The practitioner may then do Internet searches to answer this question.
- ➢ The practitioner would then use a spreadsheet such as MS-Excel to guess and document the kind of data and processes that may be used by such a business.
- ➢ Questions and gaps in knowledge must be noted down.
- ➢ This would be the pre-prototype which would be made only for the purpose of self-preparation.

This would have the effect of preparing the practitioner for the first visit.

3.2 First prototype

During the first visit and initial interviews, the practitioner would improve his understanding of the data involved and the processes.

- ➢ The practitioner would take a look at current practices of recording data, e.g. in ledger books, etc.
- ➢ Ask about business information needs and how the data is used.
- ➢ The first prototype would essentially mimic the data recording formats used by the business.
- ➢ The first prototype would be given to the user-client. The user-client would be asked to enter real data in the same manner as done in the ledger books.

The user-client should be invited to enter data in the spreadsheet in the presence of the practitioner. Once the user-client feels comfortable, he/she would be requested to continue entering data as he/she does in the ledgers. This way the user-client would become comfortable with the data entry process and also be in a position to give feedback on the use of each data item. As the user-client uses the first prototype, the following benefits may be derived.

- ➢ The user-client would start to understand how his/her data is stored in the computer.
- ➢ This would encourage user-client participation.
- ➢ The user-client would be in a position to give feedback on individual data items.

3.3 Second Prototype

With the feedback from the user-client, the data items would be improved. During the second visit he/she would be asked about the reporting requirements from this data.

- ➢ The second prototype would have buttons that produce the required reports in new sheets as required.
- ➢ The user-client would then continue entering real data and test the outputs by pressing the buttons or report producing options.

The following benefits may be derived by using the Second Prototype with buttons and reports.

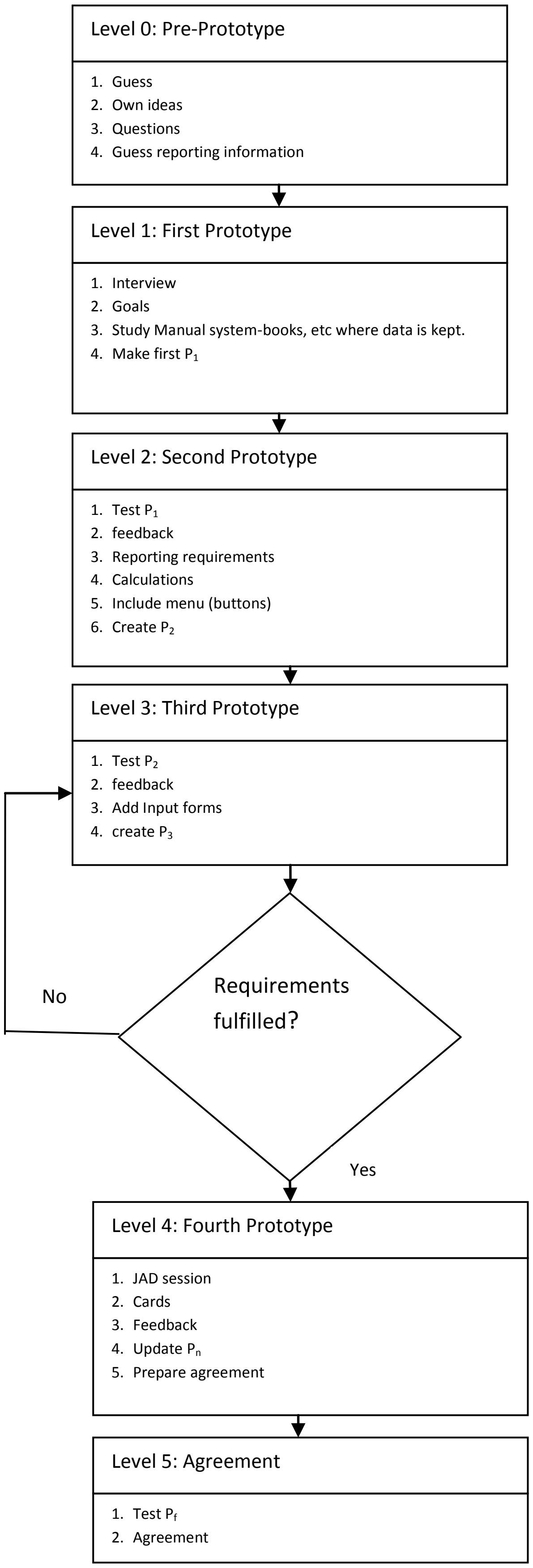


Figure 3: Business Requirement Gathering Life Cycle

➢ The user-client would be able to see what processing takes place when the buttons are pressed.

➢ He/she would be immediately able to check the output and confirm the correctness of the calculations done.

➢ He/she would be able to give feedback to improve the output reports from the prototype.

➢ The user-client would continue to use the second prototype over a periodic reporting cycle, e.g. in case of simple accounts, continue till daily, weekly and/or monthly closing.

➢ The feedback given by the user-client would be incorporated in the third prototype.

3.4 Third Prototype

In addition to the improvements incorporated from the feedback given on the second prototype, the third prototype will contain data-entry forms which the user-client will use.

➢ The third prototype will have data-entry forms.

➢ The user-client will use the third prototype with the new data-entry forms and comment on their ease of use and usability.

➢ With the feedback, the data-entry forms and messages can be improved.

The benefits of the using the third prototype:

➢ The user-client can give feedback on the usability and user-friendliness of the data-entry forms and screens.

➢ Using all the changes required and requested by the user-client or users, the fourth prototype can be prepared.

➢ At this point the Business Requirements, User Requirements and Business Rules can be finalized. Whatever accompanying documentation is required should now be prepared.

3.5 Fourth prototype

At this point, the feedback of all the stakeholders involved in the use of the software as well as those who would benefit from the use of the software should be consulted. This should be done in a participatory meeting with all the stakeholders. At the meeting, the prototype should be demonstrated and the stakeholders should also be allowed to use the software.

➢ Organize a JAD [3] workshop session with all stakeholders to get feedback on the Business Requirements, User Requirements and Business Rules.

➢ Elicit opinion using anonymous cards.

➢ Display all the cards on a pin-board get the stakeholders to group and explain their grouped ideas. Get the users to finalize their ideas.

➢ If necessary, prepare the fifth and final prototype.

Benefits of involving all the stakeholders including top management in a JAD session to give feedback using cards:

➢ Cards will allow all users and even culturally shy persons to give their anonymous feedback in the presence of their bosses.

➢ The agreement can be prepared based on the finally approved working prototype.

➢ The practitioner can add all the other parts of Requirements Engineering as shown in Figure 1 and prepare the software implementation agreement with platform, security, timelines and costs.

4. CASE STUDY

Using this methodology the first and second authors were assigned to go through the Requirement Gathering Life Cycle as shown in Figure 3 for a real company that we shall call ABC Company for anonymity. ABC Company assembles desktop PCs and delivers them to retail agents. Between the assembly location and delivery arrangements, monitoring of delivery was poor and customers complained frequently. When customers phoned in to find the status of their order, the office had to send someone to the assembly plant to find the status of progress. Three local software companies had gathered requirements and produced software. However, the customized software was not able to give the required delivery status. As a result, ABC did not have the required software for their PC delivery status.

To get an idea of what data may be involved in delivery management, we first made a pre-prototype. This helped prepare for the first interview. After visiting the company we found that three development teams had already individually tried to develop this software but they were not successful. We analyzed the reasons of the failure and decided that the main problem was in the requirement gathering phase. Basically all three parties did the same thing. They met the client and took notes of the requirements and designed and built the software. None of the customized software packages were able to solve the 'delivery status' problem.

So we first concentrated on the requirement gathering phase and implemented the Business Requirement Gathering methodology shown in Figure 3. One of the main things we found was involving the 'boss' in using the prototype and giving feedback was difficult. However, when the boss did sit down, he finally discovered that his own ledgers did not have the data required to work out the delivery status. The data items needed to track delivery were not recorded properly in the ledgers. While using the prototype, he was able to see the problem and helped set the business rule that would allow proper tracking. After getting approval of the final prototype, we developed a successful web-based system based on the prototype. Both the assembly location and delivery locations were able to use the same software and track deliveries successfully. We now explain the steps we took to finally deliver a successful delivery monitoring software for ABC Company.

Step 01:

We were assigned to develop a PC delivery management system for ABC Company. Based on our own thoughts and study we developed a pre-prototype and prepared some questions for the client to clarify their needs and then fixed a date to visit the client.

Step 02:

We visited the 'boss' of ABC Company and asked him to explain what he wanted. He explained that he regularly wants to know the latest assembly status of PCs that were ear-marked to be delivered. So whenever a customer calls, one could respond accurately with estimated delivery time. We asked several questions to understand his requirements but sometimes he failed to satisfy us with his answers. After getting the basic business requirements from the boss, we set out to study the organization. Basically there were two different sections of that company. One section receives parts for assembly and PC orders. The second section packages the PCs and was responsible for delivery. Due to space constraints the two sections were situated in different places. So the first section would essentially indicate progress of PC assembly while the second section would confirm delivery to help the boss know the status of any order.

Our first prototype looked exactly the register book he maintained. It had the same columns and rows for recording assembly data for each PC. He immediately understood the arrangement of the prototype and gave us feedback. He admitted that his requirements had changed. He wanted to search using the PC serial number or order number. So to the first sheet we added a search form. The search form is shown below:

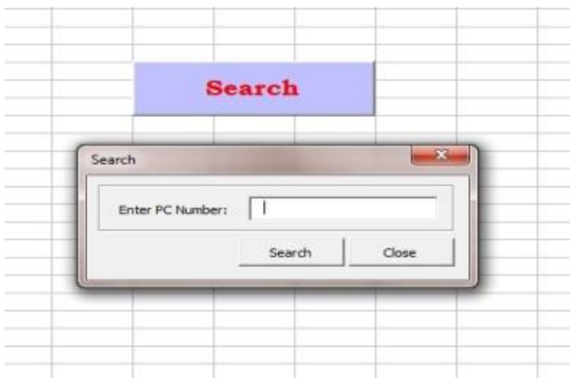


Figure 4: Search form added to first Prototype

Step 03:

After a few days we visited the boss again. This was after real data was entered on a daily basis in the spreadsheet. In the prototype he manually entered the PC number to view the status of the PC but it created some problems. Then based on the boss's advice, we added some input parameters find the desired output. In second prototype could now choose the PC Number or ID from a drop down list rather than manually enter the number to view the status. We also added some automatic user requirement features as shown in Table 1.

Input parameter	User Requirement
PC number	Auto generated
complain ID	User entered
Date	Auto generated

Based on the feedback from the boss we updated the first prototype and called it the Second Prototype.

Step 04:

In next meeting we showed the second updated prototype to the boss and asked him to check the prototype with real data that had been entered. In this prototype we added data items to get the desired output. In the previous three meetings the boss had omitted to tell us how a client ordered a PC. After using second prototype he realized some different things that he had never told us before. We clearly understood that the boss changed the requirements every time he used the updated prototype. This indicated that the boss never really gave time to understand his own data requirements. We then updated the prototype based on the changes required by the manager. The Second Prototype was then changed to the Third Prototype.

The Third Prototype now had buttons to produce the reports required by the boss.

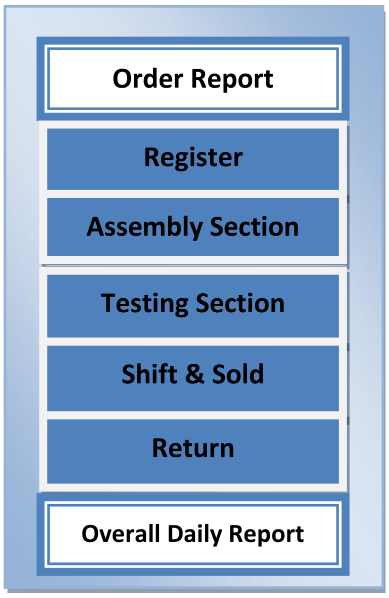


Figure 6: Third Prototype with buttons for producing reports required by the boss

Step 05:

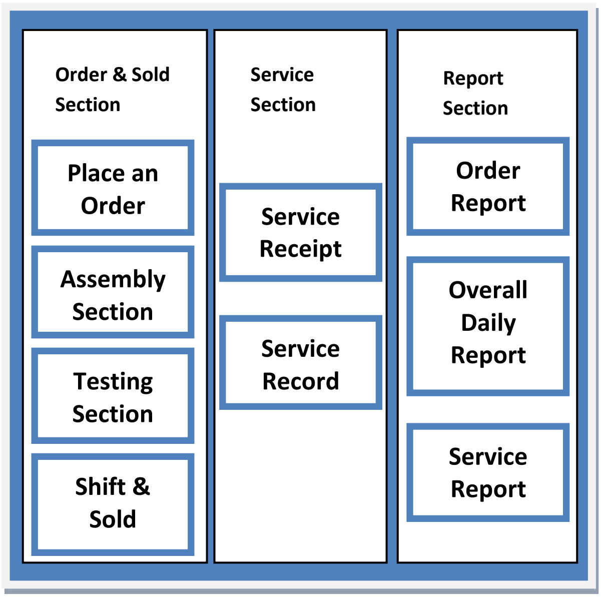
The period between third and fourth prototypes was long. We visited the company several times during that period. Every time the requirement was changed after using the prototype. Finally, we updated the prototype to

the point that satisfied the Business Requirements of the boss. We generated different reports called order, repair, assembly, service receipt, service record, and some other reports. When we added each report we visited the boss and asked him to test using the real data input by his two sections. Every time he used it by using the real data he gave feedback; we changed it according to his need. After the fourth prototype changes were made at least four times. The manager was finally satisfied to see his desired output. The outputs were completely different from his first required outputs.

Table 1: User Requirement features added

4.1 JAD Workshop Session

After confirmation with the boss we organized a JAD (Joint Application Design) session with all the employees of the company. We distributed 3”x5” cards. We described the total system and demonstrated the prototype in front of them using a multimedia projector. After that we asked the employees to write their opinion on the cards. We then got two volunteers to organize the cards based on category of comments and tried to understand and solve the issues mentioned. We received some suggestions from them and then finally updated the prototype. After a successful JAD session the boss of the company was highly impressed and gave us an approval letter to do develop the actual software. After the suggestions received from the JAD session, the user requirements of all the employees were received and the final interface looked as shown below in Figure 7.



As can be seen, the user interface was eventually divided three different sections as a result of the JAD session. The three groups helped allowed the users to exactly know which groups applied to them.

5. FINDINGS

While doing the project we faced a number of problems:

➢ The main problem was that boss seemed to know what he wanted, but did not organize his own manual registers to deliver what he wanted. The prototype helped the boss visualize and hence organize the data to deliver what he wanted. In time we realized how important the live prototype method was – it helped the boss work out and specify his own business requirements. The live prototype helped the boss see which data items were necessary to deliver the information he needed for running his business.

➢ As for implementing the concept of requirement gathering using a prototype was completely new for the boss, we faced some problems. At first he did not understand the concept of using a prototype. The prototype was in a spreadsheet platform. So the design was not attractive. For this reason the boss was initially unhappy with the design. It took time to understand that this was only a demo! Final design will be different from it. In the beginning the boss did not properly cooperate with us. But after understanding the concept he and his organization helped us a lot.

6. BENEFITS AND DISCUSSIONS

Requirement gathering prototype helped both the client and the practitioner in several ways.

➢ The model successfully collected the requirements from the client.

➢ It minimized the communication gap between the user-client and the practitioner.

➢ The model broke down the requirement gathering process into several parts so that client was able to specify his own requirements as he developed understanding of what was required.

➢ It was easy to change the Excel prototype. Often changes were done immediately at user-client site.

➢ The boss had not bothered to design his manual system to give the required results. Any software system based on the previous manual system would be bound to fail.

➢ The model is clearly defined, easy to understand and may also be changed around to suit the needs of the user-client.

➢ The prototype acts like a 'live' Systems Requirement Specification Document.

➢ If the team is changed during the development process then new team will understand the requirements easily from the prototype without disturbing the user-client.

➢ Compared to the review [4] of modern rapid prototyping software packages, the use of spreadsheets allows the user-client to visually see the data and processing directly and thereby give appropriate feedback to finalize business and user requirements.

7. LIVE PROTOTYPE AND AGILE METHODOLOGY

The live prototyping approach supports the main principles of agile methodology used to elicit user requirements, e.g. on-site and co-located user-client, frequent short releases, flexible to changing requirements, iterative development etc [5, 6]. For implementing interaction, the agile methodology suggests user-stories or use-cases for the high-level requirement elicitation or extraction that are prioritized according to their business value [5]. The basic components of a user story include Cards (physical medium), Conversation (discussion surrounding stories) and Confirmation (tests and validation that verify story completion) [5]. The user stories, written on 3"x5" index cards are expected to be short, to the point and validation-centric. A well-written user story demand to follow the INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable) model and a certain template (Who, What, Why) [5]. Now to meet up these conditions, only manual process that is the existing agile way to fix requirements may become tough and confusing for both parties (user-client and developer). Live prototyping tool might be useful here to make user-client more comfortable whenever he/she is unsure and find the whole process complex. A live demonstration of business processes is more worthy than using thousand words in conversation.

In the Agile methodology, criteria for user stories make user-clients write acceptance tests that are to be transformed into unit tests by the developers [6]. Acceptance criteria include functionality that the system will perform, interface look and feel, necessary documentation etc [5]. Again, the live prototyping tool may play a useful role in terms of requirement confirmation.

SCRUM, one of the agile methodologies treats the requirements like a prioritized task [5, 11, 12]. It freezes the requirements for the current iteration to provide a level of stability for the developers. In Scrum, work is expressed in the backlog as user stories and to accomplish this, it includes several types of meetings (Sprint Planning Meeting, Daily Scrum meeting, Sprint Review Meeting etc.) with user-client [11, 12]. The live prototyping tool may be used here also.

Agile methodology gives more value to customer collaboration over contract negotiation where it surely emphasizes customer involvement in requirement gathering process by taking his feedback [9] that our proposed tool also demands. In this development methodology, the client is mostly present in the entire development process to fix goals, freeze requirements and create product backlogs. The live prototype provides a structured practical tool to make the requirement elicitation process easier. The tool is intended to be used for a more efficient RE.

8. CONCLUSION

For small businesses which have not taken the time to efficiently design their own business requirements, Requirement Gathering using a Live Prototyping method may prove to be useful. The prototype can be changed rapidly and allows the user-client to freeze his/her business requirements easily.

9. REFERENCES

[1] Westfall, L. (2005). Software Requirements Engineering: What, Why, Who, When, and How, *Software Quality Professional*, Proquest, Vol. 7, Issue4, 2005, Pages: 17-26. Retrieved April, 2013 from http://www.westfallteam.com/Papers/The_Why_What_Who_When_and_How_Of_Software_Requirements.pdf

[2] Wiegers, K. E. (2000). "When Telepathy Won't Do: Requirements Engineering Key Practices," *Cutter IT Journal*, vol. 13, no. 5 (May 2000). Retrieved April, 2013 from http://www.processimpact.com./articles/telepathy.pdf

[3] Kuchmistaya, S. B. (2001). Incorporation of Joint Application Design (JAD) in Systems Requirement Determination. Retrieved April 2013, from http://www.umsl.edu/~sauterv/analysis/488_f01_papers/kuchmistaya.htm

[4] Nayak, A. & Dash, J. (2011). Requirement Analysis and Rapid Prototyping. Retrieved April, 2013 from http://www.slideshare.net/conviction/requirement-gathering-rapid-prototyping

[5] Kavita & Sunitha (2011). "Requirement Gathering for Small Projects Using Agile Methods", IJCA Special Issue on "Computational Science - New Dimensions & Perspectives, Special Issue.

[6] Bose, Kurhekar & Ghoshal (2008). Agile Methodology in Requirements Engineering.

[7] Pressman, R. S. (2005), Software Engineering: A Practitioner's Approach, Sixth Edition, McGraw-Hill International Edition.

[8] K. Kautz, H.W. Hansen & K. Thaysen. “Applying and Adjusting a Software Process Improvement Model in Practice: The Use of the IDEAL Model in a Small Software Enterpris”e, 22nd Int’l. Conf. on Software Engineering, pp. 626-633.

[9] Cohn, M (2009), Succeeding with Agile: Software Development using Scrum.

[10] Sommerville. Software Engineering, AddisonWesley.

[11] Elizabeth Woodward, 2010. A Practical Guide to Distributed Scrum, IBM Press

[12] The Enterprise and SCRUM, 2007, Microsoft Press,